REMARKS

Applicants respectfully request the Examiner's reconsideration of the present application as amended.

Claims 1, 3-20 are pending in the present application.

Claim 15 is rejected because of informalities.

Claims 6 and 17-20 are rejected under 35 U.S.C. §112, first paragraph.

Claims 6 and 19 are rejected under 35 U.S.C. §112, second paragraph.

Claims 1, 3-4, 7-8 and 10-16 are rejected under 35 U.S.C. §102(b) as being unpatenable over a publication entitled "Partitioning a Lenient Parallel Language into Sequential Threads" by Sangho Ha, Sangyong Han, and Heunghwan Kim, published in 1995 ("Ha").

Claims 5 and 9 are rejected under 35 U.S.C. §103(a) as being unpatentable over Ha in view of U.S. Patent No. 7,768,594 ("Blelloch").

Claim 3-4, and 7-8 have been canceled.

Claims 1, 5-6, 9, 15, and 19 have been amended.

Claims 21-22 have been added.

Support for the new and amended claims may be found at paragraphs [0026]-[0028] of the specification, Figures 2 and 3 of the drawings, and Claims 1-20 as originally filed. No new matter has been added.

Claim 15 is objected to because of informalities.

Claim 15 has been amended according to the Office's suggestions.

Applicants submit that in view of the amendment to Claim 15, the objection to the claim has been overcome.

Claims 6 and 17-20 are rejected under 35 U.S.C. §112, first paragraph, as containing subject matter which was not described in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

Specifically, the Office states in part the following.

> Claim 6 recites "if a compute weight of the upstream stage exceeds a predetermined value". While the specification refers repeatedly to a "computed weight" nowhere do the applicants disclose what a "computed weight" is or, more importantly, how it would be calculated. Further, the term "weight" does not appear to be a term used in the relevant arts. Accordingly those of ordinary skill in the art would not have been enabled to calculate a "compute weight of the upstream stage" in accordance with the applicants disclosed embodiments without undue experimentation.

(4/12/2011 Office Action, pp. 8-5) (Emphasis Added).

Applicants respectfully disagree. For example, the term "weight" has been used in the art as a unit associated with compute time. See "Fair Scheduler Guide", p. 2, by the Apache Software, Foundation, 2008. The term "weight" has also bee used to refer to a parameter associated with an amount of CPU cycles. See "Memory and CPU allocation in Xen environments: Optimizing performance", p. 5, Sander van Vugt, 2007.

Applicants submit that in view of the explanation above, the rejection of claims 6, and 17-20 under 35 U.S.C. §112, first paragraph have been overcome.

Claims 6 and 19 are rejected under 35 USC § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which Applicant regards as the invention.

Applicants submit that with reference to Claim 6, "the first new number" has been amended to "the new number". Furthermore, Applicants submit that "the new number of desired upstream nodes" implies a different number of desired upstream nodes and a different set of nodes would be assigned to the upstream stage.

Claim 19 has been amended to change its dependency from Claim 16 to Claim 17.

Applicants submit that in view of the amendments, the rejection to Claims 6 and 19 under 35 U.S.C. §112, second paragraph have been overcome.

Claims 1, and 3-20 are rejected under 35 U.S.C. §102(b) and §103(a) as being unpatentable over Ha in view of Blelloch.

It is submitted that Ha and Blelloch do not render claims 1, 5-6, and 9-22 unpatentable under 35 U.S.C. §102(b) and §103(a).

Ha includes a disclosure of a thread formation scheme to produce efficient sequential threads from programs written in a lenient parallel language. This scheme features graph partitioning based on only long latency instructions, combination of multiple switches and merges introducing a generalized switch and merge, thread merging, and redundant arc elimination using thread precedence relations (see Ha Abstract).

Blelloch includes a disclosure of a method of parallel processing by determining sequential ordering of tasks for processing, assigning priorities to the tasks available on the basis of the sequential ordering, selecting a number of tasks greater than a total number of available parallel processing elements from all available tasks having the highest priorities, partitioning the selected tasks into a number of groups equal to the available number of parallel processing elements, and executing the tasks in the parallel processing elements (see Blelloch Abstract).

Ha and Blelloch do not teach or suggest partitioning instructions in the code among a plurality of processors based on memory access latency associated with the instructions by partitioning memory access dependence chains into an upstream stage by assigning a first number of desired upstream nodes to the upstream stage, and assigning instructions in the code on which the first number of desired upstream nodes are dependent to the upstream stage, and partitioning the instructions among the plurality of processors by partitioning the memory access dependence

chains into a downstream stage by assigning a last number of desired downstream nodes to the downstream stage, and assigning instructions in the code which are dependent on the last number of desired downstream nodes to the downstream stage.

In the Office Action mailed 4/12/2011, the Office states in part the following.

> Claims 4 and 15... Ha discloses partitioning instructions comprising partitioning a memory access dependence chain into an upstream stage by assigning a first number of desired upstream nodes to the upstream stage, and assigning instructions in the code on which the first number of desired upstream nodes are dependent to the upstream stage (pg. 85, col. 2, Section 4.1, 2nd par. "We partition the DAVRID graphs by -- using dependence set developed by Iannucci[4]", pg. 85 col. 1 1st partial par. "Note that arcs in the DAVID graphs ... denote ... dependencies between nodes").
>
> . . .
>
> Claim 8... Ha discloses partitioning the memory access dependence chain into the downstream stage comprises:
> assigning a last number of desired downstream nodes to the downstream stage (pg. 85, col. 2, Section 4.1, 2nd par. "We partition the David graphs by cutting remote arcs logically"); and
> assigning instructions in the code which are dependent on the last number of downstream nodes to the downstream stage (pg. 85, col. 2 Section 4.1, 2nd par. "We partition the DAVRID graphs by -- using dependence set developed by Iannucci[4]", pg. 85 col. 1 1st partial par. "Note that arcs in the DAVID graphs ... denote ... dependencies between nodes").

(4/12/2011 Office Action, pp. 11-12)

Applicants respectfully disagree. Applicants submit that "cutting remote arcs logically" is not equivalent to assigning a first number of desired upstream nodes to an upstream stage, and assigning instructions in the code on which the first number of desired upstream nodes are dependent on the upstream stage, as claimed. Furthermore, "cutting remote arcs logically" is not equivalent to assigning a last number of desired downstream nodes to the downstream stage, and assigning instructions in the code which are dependent on the last number of desired downstream nodes to the downstream stage. Applicants submit that Ha does not make reference to

upstream/downstream nodes, upstream/downstream stages, or first/last number of desired upstream/downstream nodes dependent on the upstream/downstream stage.

Blelloch does not cure the deficiencies of Ha.

In contrast, claim 1, as amended states

> A method of compiling code, comprising:
> partitioning instructions in the code among a plurality of processors based on memory access latency associated with the instructions by partitioning memory access dependence chains into an upstream stage by assigning a first number of desired upstream nodes to the upstream stage, and assigning instructions in the code on which the first number of desired upstream nodes are dependent to the upstream stage; and
> partitioning the instructions among the plurality of processors by partitioning the memory access dependence chains into a downstream stage by assigning a last number of desired downstream nodes to the downstream stage, and assigning instructions in the code which are dependent on the last number of desired downstream nodes to the downstream stage.

(Claim 1, as Amended) (Emphasis Added).

Given that claims 5, 6, and 9-12 depend from Claims 1, it is likewise submitted that claims 5, 6, and 9-12 are also patentable under 35 U.S.C. §102(b) and §103(a) over Ha and Blelloch by virtue of their dependency.

Ha and Blelloch also do not teach or suggest partitioning instructions in code into a plurality of pipeline stages to be executed in parallel by a plurality of processors based on memory access latency associated with the instructions.

On the contrary, Ha does not make any reference to "pipeline stages" or "pipelining" anywhere in its text. Neither Ha's Abstract, page 83, column 1, nor page 84 column 1, cited by the Office, describes pipeline stages. Applicants respectfully requests that the Office provide the location in Ha where it believes it discloses partitioning instructions in code into a plurality of pipeline stages, as claimed.

Blelloch also does not make any references to "pipeline stages" or "pipelining" anywhere in its text. Blelloch fails to cure the deficiencies of Ha.

In contrast, claim 13 states

> An article of manufacture comprising a non-transitory machine accessible medium including sequences of instructions, the sequences of instructions including instructions which when executed cause the machine to perform:
> partitioning instructions in code into a plurality of pipeline stages to be executed in parallel by a plurality of processors based on memory access latency associated with the instructions.

(Claim 13) (Emphasis Added).

Claim 16 includes similar limitations. Given that claims 14-15, and 17-22 depend from claims 13 and 16, it is likewise submitted that claims 14-15, and 17-22 are also patentable under 35 U.S.C. §102(b) and §103(a) over Ha and Blelloch by virtue of their dependency.

Ha and Blelloch also do not teach or suggest partitioning a memory access dependence chain into an upstream stage by assigning a first number of desired upstream nodes to the upstream stage, wherein the number of desired upstream nodes is the length of the memory access dependence chain divided by a pipelining degree.

The Office states in part the following.

> Claim 5: The rejection of claim 4 is incorporated; further Ha does not disclose the number of desired upstream nodes is the length of the memory access dependence chain divided by a pipelining degree.
>
> Blelloch teaches partitioning code such that the number of desired upstream nodes is the length of the memory access dependence chain divided by a pipelining degree (col. 4, lines 19-22 "In step 610, the assignment manager AM1 partitions the N selected tasks to p groups of size approx (N/p) each, where p is the number of processing elements PE1").

(9/22/2010 Office Action, p. 10) (Emphasis Added).

Applicants submit that Blelloch fails to cure the deficiencies of Ha.

Firstly, Blelloch discloses partitioning N selected tasks, to p groups. However, N is described by Blelloch as being a number of available tasks which have the highest assigned priority, not a number reflecting all available tasks such as those found in a memory access

dependence chain (see Blelloch column 4, lines 13-17). A number of available tasks which have the highest assigned priority is not equivalent to a length of a memory access dependence chain.

Blelloch specifically states the following.

> In step 607, the assignment manager AM1 selects some number N of available tasks which have the highest assigned priority, where N is typically, but not necessarily more than the number of processing elements and less than the maximum possible available tasks.

(Blelloch column 4, lines 13-17) (Emphasis Added).

Secondly, Blelloch fails to make any reference to "memory access" or "memory access dependence chain" anywhere in its specification. So regardless of what N reflects, Ha fails to teach or suggest a number of desired upstream nodes defined as a length of a memory access dependence chain divided by a pipelining degree, as claimed.

In contrast, claim 5 states

> The method of Claim 4, wherein the number of desired upstream nodes is the length of the memory access dependence chain divided by a pipelining degree.

(Claim 5) (Emphasis Added).

In view of the arguments set forth herein, it is respectfully submitted that the applicable objections and rejections and have been overcome. Accordingly, it is respectfully submitted that claims 1, 5-6, and 9-22 should be found to be in condition for allowance.

The Examiner is invited to telephone Applicants' attorney (217-377-2500) to facilitate prosecution of this application.

If necessary, please charge any additional fees or credit overpayment to Deposit Account No. 50-4238.

Respectfully submitted,
Customer Number: 45512
217-377-2500

Date July 12, 2011

By */Lawrence M. Cho/*
Lawrence M. Cho
Attorney for Applicants
Registration No. 39,942

CERTIFICATE UNDER 37 CFR 1.8: The undersigned hereby certifies that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: Mail Stop Amendment, Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on this 12th day of July, 2011.

/Keith Lodermeier/ July 12, 2011

Keith Lodermeier

Fair Scheduler Guide

Table of contents

1. Purpose

This document describes the Fair Scheduler, a pluggable Map/Reduce scheduler for Hadoop which provides a way to share large clusters.

2. Introduction

Fair scheduling is a method of assigning resources to jobs such that all jobs get, on average, an equal share of resources over time. When there is a single job running, that job uses the entire cluster. When other jobs are submitted, tasks slots that free up are assigned to the new jobs, so that each job gets roughly the same amount of CPU time. Unlike the default Hadoop scheduler, which forms a queue of jobs, this lets short jobs finish in reasonable time while not starving long jobs. It is also a reasonable way to share a cluster between a number of users. Finally, fair sharing can also work with job priorities - the priorities are used as weights to determine the fraction of total compute time that each job should get.

The scheduler actually organizes jobs further into "pools", and shares resources fairly between these pools. By default, there is a separate pool for each user, so that each user gets the same share of the cluster no matter how many jobs they submit. However, it is also possible to set a job's pool based on the user's Unix group or any other jobconf property, such as the queue name property used by Capacity Scheduler. Within each pool, fair sharing is used to share capacity between the running jobs. Pools can also be given weights to share the cluster non-proportionally in the config file.

In addition to providing fair sharing, the Fair Scheduler allows assigning guaranteed minimum shares to pools, which is useful for ensuring that certain users, groups or production applications always get sufficient resources. When a pool contains jobs, it gets at least its minimum share, but when the pool does not need its full guaranteed share, the excess is split between other running jobs. This lets the scheduler guarantee capacity for pools while utilizing resources efficiently when these pools don't contain jobs.

The Fair Scheduler lets all jobs run by default, but it is also possible to limit the number of running jobs per user and per pool through the config file. This can be useful when a user must submit hundreds of jobs at once, or in general to improve performance if running too many jobs at once would cause too much intermediate data to be created or too much context-switching. Limiting the jobs does not cause any subsequently submitted jobs to fail, only to wait in the sheduler's queue until some of the user's earlier jobs finish. Jobs to run from each user/pool are chosen in order of priority and then submit time, as in the default FIFO scheduler in Hadoop.

Finally, the fair scheduler provides several extension points where the basic functionality can

be extended. For example, the weight calculation can be modified to give a priority boost to new jobs, implementing a "shortest job first" policy which reduces response times for interactive jobs even further.

3. Installation

To run the fair scheduler in your Hadoop installation, you need to put it on the CLASSPATH. The easiest way is to copy the *hadoop-*-fairscheduler.jar* from *HADOOP_HOME/contrib/fairscheduler* to *HADOOP_HOME/lib*. Alternatively you can modify *HADOOP_CLASSPATH* to include this jar, in *HADOOP_CONF_DIR/hadoop-env.sh*

In order to compile fair scheduler, from sources execute *ant package* in source folder and copy the *build/contrib/fair-scheduler/hadoop-*-fairscheduler.jar* to *HADOOP_HOME/lib*

You will also need to set the following property in the Hadoop config file *HADOOP_CONF_DIR/mapred-site.xml* to have Hadoop use the fair scheduler:

```
<property>
  <name>mapred.jobtracker.taskScheduler</name>
  <value>org.apache.hadoop.mapred.FairScheduler</value>
</property>
```

Once you restart the cluster, you can check that the fair scheduler is running by going to http://<jobtracker URL>/scheduler on the JobTracker's web UI. A "job scheduler administration" page should be visible there. This page is described in the Administration section.

4. Configuring the Fair scheduler

The following properties can be set in mapred-site.xml to configure the fair scheduler:

Name	Description
mapred.fairscheduler.allocation.file	Specifies an absolute path to an XML file which contains the allocations for each pool, as well as the per-pool and per-user limits on number of running jobs. If this property is not provided, allocations are not used. This file must be in XML format, and can contain three types of elements: • pool elements, which may contain elements for minMaps, minReduces, maxRunningJobs (limit the number of jobs from the pool to run at once),and weight (to share the cluster non-proportionally with other pools).

	• user elements, which may contain a maxRunningJobs to limit jobs. Note that by default, there is a separate pool for each user, so these may not be necessary; they are useful, however, if you create a pool per user group or manually assign jobs to pools.
	• A userMaxJobsDefault element, which sets the default running job limit for any users whose limit is not specified.
	
	Example Allocation file is listed below :
	`<?xml version="1.0"?>`
	`<allocations>`
	`<pool name="sample_pool">`
	`<minMaps>5</minMaps>`
	`<minReduces>5</minReduces>`
	`<weight>2.0</weight>`
	`</pool>`
	`<user name="sample_user">`
	`<maxRunningJobs>6</maxRunningJobs>`
	`</user>`
	`<userMaxJobsDefault>3</userMaxJobsDefault>`
	`</allocations>`
	This example creates a pool sample_pool with a guarantee of 5 map slots and 5 reduce slots. The pool also has a weight of 2.0, meaning it has a 2x higher share of the cluster than other pools (the default weight is 1). Finally, the example limits the number of running jobs per user to 3, except for sample_user, who can run 6 jobs concurrently. Any pool not defined in the allocations file will have no guaranteed capacity and a weight of 1.0. Also, any pool or user with no max running jobs set in the file will be allowed to run an unlimited number of jobs.

mapred.fairscheduler.assignmultiple	Allows the scheduler to assign both a map task and a reduce task on each heartbeat, which improves cluster throughput when there are many small tasks to run. Boolean value, default: false.
mapred.fairscheduler.sizebasedweight	Take into account job sizes in calculating their weights for fair sharing.By default, weights are only based on job priorities. Setting this flag to true will make them based on the size of the job (number of tasks needed) as well,though not linearly (the weight will be proportional to the log of the number of tasks needed). This lets larger

	jobs get larger fair shares while still providing enough of a share to small jobs to let them finish fast. Boolean value, default: false.
mapred.fairscheduler.poolnameproperty	Specify which jobconf property is used to determine the pool that a job belongs in. String, default: user.name (i.e. one pool for each user). Some other useful values to set this to are: • group.name (to create a pool per Unix group). • mapred.job.queue.name (the same property as the queue name in Capacity Scheduler).
mapred.fairscheduler.weightadjuster	An extensibility point that lets you specify a class to adjust the weights of running jobs. This class should implement the *WeightAdjuster* interface. There is currently one example implementation - *NewJobWeightBooster*, which increases the weight of jobs for the first 5 minutes of their lifetime to let short jobs finish faster. To use it, set the weightadjuster property to the full class name, `org.apache.hadoop.mapred.NewJobWeightBooster` NewJobWeightBooster itself provides two parameters for setting the duration and boost factor. 1. *mapred.newjobweightbooster.factor* Factor by which new jobs weight should be boosted. Default is 3 2. *mapred.newjobweightbooster.duration* Duration in milliseconds, default 300000 for 5 minutes
mapred.fairscheduler.loadmanager	An extensibility point that lets you specify a class that determines how many maps and reduces can run on a given TaskTracker. This class should implement the LoadManager interface. By default the task caps in the Hadoop config file are used, but this option could be used to make the load based on available memory and CPU utilization for example.
mapred.fairscheduler.taskselector:	An extensibility point that lets you specify a class that determines which task from within a job to launch on a given tracker. This can be used to change either the locality policy (e.g. keep some jobs within a particular rack) or the speculative execution algorithm (select when to launch speculative tasks). The default implementation uses Hadoop's default algorithms from

	JobInProgress.

5. Administration

The fair scheduler provides support for administration at runtime through two mechanisms:

1. It is possible to modify pools' allocations and user and pool running job limits at runtime by editing the allocation config file. The scheduler will reload this file 10-15 seconds after it sees that it was modified.
2. Current jobs, pools, and fair shares can be examined through the JobTracker's web interface, at http://<jobtracker URL>/scheduler. On this interface, it is also possible to modify jobs' priorities or move jobs from one pool to another and see the effects on the fair shares (this requires JavaScript).

The following fields can be seen for each job on the web interface:

- *Submitted* - Date and time job was submitted.
- *JobID, User, Name* - Job identifiers as on the standard web UI.
- *Pool* - Current pool of job. Select another value to move job to another pool.
- *Priority* - Current priority. Select another value to change the job's priority
- *Maps/Reduces Finished*: Number of tasks finished / total tasks.
- *Maps/Reduces Running*: Tasks currently running.
- *Map/Reduce Fair Share*: The average number of task slots that this job should have at any given time according to fair sharing. The actual number of tasks will go up and down depending on how much compute time the job has had, but on average it will get its fair share amount.

In addition, it is possible to turn on an "advanced" view for the web UI, by going to http://<jobtracker URL>/scheduler?advanced. This view shows four more columns used for calculations internally:

- *Maps/Reduce Weight*: Weight of the job in the fair sharing calculations. This depends on priority and potentially also on job size and job age if the *sizebasedweight* and *NewJobWeightBooster* are enabled.
- *Map/Reduce Deficit*: The job's scheduling deficit in machine- seconds - the amount of resources it should have gotten according to its fair share, minus how many it actually got. Positive deficit means the job will be scheduled again in the near future because it needs to catch up to its fair share. The scheduler schedules jobs with higher deficit ahead of others. Please see the Implementation section of this document for details.

6. Implementation

There are two aspects to implementing fair scheduling: Calculating each job's fair share, and

choosing which job to run when a task slot becomes available.

To select jobs to run, the scheduler then keeps track of a "deficit" for each job - the difference between the amount of compute time it should have gotten on an ideal scheduler, and the amount of compute time it actually got. This is a measure of how "unfair" we've been to the job. Every few hundred milliseconds, the scheduler updates the deficit of each job by looking at how many tasks each job had running during this interval vs. its fair share. Whenever a task slot becomes available, it is assigned to the job with the highest deficit. There is one exception - if there were one or more jobs who were not meeting their pool capacity guarantees, we only choose among these "needy" jobs (based again on their deficit), to ensure that the scheduler meets pool guarantees as soon as possible.

The fair shares are calculated by dividing the capacity of the cluster among runnable jobs according to a "weight" for each job. By default the weight is based on priority, with each level of priority having 2x higher weight than the next (for example, VERY_HIGH has 4x the weight of NORMAL). However, weights can also be based on job sizes and ages, as described in the Configuring section. For jobs that are in a pool, fair shares also take into account the minimum guarantee for that pool. This capacity is divided among the jobs in that pool according again to their weights.

Finally, when limits on a user's running jobs or a pool's running jobs are in place, we choose which jobs get to run by sorting all jobs in order of priority and then submit time, as in the standard Hadoop scheduler. Any jobs that fall after the user/pool's limit in this ordering are queued up and wait idle until they can be run. During this time, they are ignored from the fair sharing calculations and do not gain or lose deficit (their fair share is set to zero).

- Server virtualization strategies and use cases

  Desktop virtualization, Lab, test and development, Cloud computing and virtualization strategies, Data center consolidation, Backup, disaster recovery and business continuity

  Benefits of server virtualization

  Server consolidation and improved resource utilization, Green data center: Reducing power consumption with virtualization, Reducing IT costs with server virtualization, Improving server management with virtualization

  Challenges of server virtualization

  Virtualization costs, licensing and support issues, Virtualization security and patch management, Downtime and data loss in virtualized environments, Preventing virtual machine sprawl

- **Hot Topics**
  - Vendor selection
  - Virtual server backup and storage
  - VM performance management

- Tutorials
  - **Advice & Tutorials**
    - Virtualization Decisions 2010 Purchasing Intentions Survey
    - Hyper-V vs. VMware guide
    - Private cloud computing tutorial: An introduction to private clouds
    - Control VM sprawl in your virtual server infrastructure
    - Virtualization Explained: Virtualization definitions you need to know
    - vSphere and vCenter licensing and pricing explained -- a VMWare license guide
  - **Technology Dictionary**
    - Find definitions and links to technical resources
    - Powered by WhatIs.com
- Expert Advice
  - **Tips**
    - Cloud management with Microsoft Concero: Battling VMware vCloud Director
    - Data deduplication reduces storage requirements: Law firm case study
    - DMZ design: Bringing virtualization and security admins together
    - View All Tips
  - **Answers**
    - VMware acquisitions: Moving beyond virtualization
    - IT career advice for graduates
    - Software licensing models need to get with the times
    - View All Answers

Memory and CPU allocation in Xen environments: Optimizing performance

Sander van Vugt, contributor

Managing hardware in a Xen environment doesn't stop after telling a virtual machine what PCI devices it can use. In a paravirtual environment, it is also possible to change memory and CPU allocations dynamically. Do so, and you'll maximize virtual machine performance. In this article, you'll learn everything about it.

When your physical server boots, all memory by default is allocated to the dom0. When other virtual machines are started, they can take from that memory. If the virtual machine is used in full virtualization mode, there is no way that the hypervisor can talk to the virtualized kernel and you can't change the current memory allocation. If, however, the virtual machine is in paravirtualization, the Xen hypervisor can change the memory allocation of that machine dynamically. When doing this, you should however always make sure that a given minimum amount of memory stays available for the Dom0 machine, as you don't want it to run out of memory. I'd recommend to set this minimum to 512 MB.

To make this initial memory reservation for Dom0, you need to add a boot option to your kernel. The option to use is dom0_mem=, for example: dom0_mem=512M. To make this setting, open the Grub configuration file with an editor. In this file, you'll find the option that starts a Xen kernel. It may look as in the following example:

```
title XEN
 root (hd0,0)
 kernel /xen.gz
 module /vmlinuz-2.6.16.46-0.14-xen root=/dev/system/root vga=0x314
      resume=/dev/system/swap splash=silent showopts
 module /initrd-2.6.16.46-0.14.xen
```

In this configuration, on the first "module" line, add the dom0_mem option. The result may look as in the following:

```
title XEN
 root (hd0,0)
 kernel /xen.gz
 module /vmlinuz-2.6.16.46-0.14-xen root=/dev/system/root vga=0x314
      resume=/dev/system/swap splash=silent showopts dom0_mem=512M
 module /initrd-2.6.16.46-0.14.xen
```

Now that you have fixed the amount of memory that will always be available for Dom0, you can manage the memory allocation for your virtual machines. When a virtual machine starts, it normally takes the memory that it has assigned from the memory that is available to Dom0. Once allocated, the Dom0 will never get that memory back, not even when the virtual machines are all stopped. Especially for that reason, it is important that you assign a minimal amount of memory that will always be assigned to the Dom0.

To change the memory allocation for virtual machines, there are two xm commands that you can use:

xm mem-set : use this to change the current memory allocation for a virtual machine.
xm mem-max : use this to limit the maximum amount of memory that a virtual machine is allowed to use. Be aware however that the new maximum setting will be applied only after the machine has rebooted.

After changing the memory assignment, always use the xm list command to check if it has worked out the way you wanted:

Listing 1: Use xm list

- More on Citrix XenServer
- Get help from the community
- Powered by ITKnowledgeExchange.com

to check memory assignments on a regular basis

lin:~ # xm list

Name	ID	Mem	VCPUs	State	Time(s)
Domain-0	0	898	2	r----	106.1
main		512>	1		16.1
oes2cursus	1	600	1	-b---	21.7
oes2l	2	512	1	-b---	1.0
sled10		256	1		85.1
sles10		512	1		0.0
sles10-1		512	1		0.0
windows2003		256	1		2.7

Managing CPU's

As with memory, you can manage CPUs that are assigned to a virtual machine as well. If your virtual machine uses paravirtualization, it is also possible to change CPU assignments dynamically. When assigning CPUs to a virtual machine, you are not bound to the number of CPUs that are physically installed in your server. You can go beyond that if you like, but do notice that there is absolutely no performance gain at all if you do so. Something that is very useful indeed, is the possibility to pin a virtual machine to a given physical CPU. This may greatly help you improve the performance of your virtual machines. Apart from that, you can tune the CPU run queue to give one virtual machine more priority on a CPU than another virtual machine.

All runnable virtual CPUs (VCPUs) are managed by the local run queue on a physical CPU. This queue is sorted by VCPU priority. In the queue, every VCPU gets its fair share of CPU resources. The status of a VCPUs priority can have two values: It is over if it has consumed more CPU resources than it normally would be allowed to and it is under if it has not yet reached that value. If a VCPU has a current status of under, it will always come first when the scheduler next decides what VCPU to service. This even works beyond physical CPUs; if the scheduler doesn't see a virtual machine with a status of under on its current CPU, it will look at the other CPU to see if it can find a VCPU with such a status there and if it does, it will service that VCPU immediately. By doing this, all CPUs would normally get their fair amount of CPU resources.

As an administrator, you can manage the priority that a CPU would get by manipulating the weight and the cap values. The weight parameter is used to assign the amount of CPU cycles that a domain receives. Weight is relative. A VCPU with a weight of 128 would receive twice as much CPU cycles as a VCPU with a weight of 64. So use this parameter to determine which VCPU should get more, and which should get less attention. The second parameter to tune what a CPU may be doing, is the cap parameter. This parameter defines in a percentage the maximum amount of CPU cycles that a domain will receive. This is an absolute value; If it is set to 100, it means that the VCPU may consume 100% of available cycles on a physical CPU, if you set it to 50, then that would mean that the VCPU can consume never more than half of the available cycles.

In the following example command, the weight of the machine with id 3 is set to 128, and the machine is allowed to use all CPU cycles on two physical CPUs:

```
xm sched-credit -d 3 -w 128 -c 200
```

Another important task with regard to virtual CPU, is CPU allocation. By default, there is no fixed relation between a virtual CPU and a physical CPU. To improve performance, such relations can be established easily. The major benefit of this "pinning" of a VCPU to a physical CPU is that you prevent a VCPU from

floating around. Without pinning, the scheduler determines by what physical CPU a virtual CPU is serviced. If one physical CPU is busy, the virtual CPU can float to another core. In terms of performance, this is quite an expensive action. Therefore it is a good idea to pin virtual CPU's to physical CPU's at all times.

To pin a VCPU, first use the xm list command to see what your current configuration looks like. Next, use the xm vcpu-list on the domain for which you want to see CPU details. The result of this command looks as follows:

```
lin:- # xm vcpu-list 2

Name                              ID  VCPU   CPU State   Time(s) CPU Affinity

oes21                              2              0        1      r--     3693.8   any cpu
```

This command shows that the domain with ID 2 currently uses one CPU with the ID 0, which is currently allocated on physical CPU 0. To make sure that it stays like that, you can now use the following command:

```
xm vcpu-pin 2 0 1
```

If you next use the xm vcpu-list command again, you'll see that the CPU Affinity has now changed from "any cpu" to CPU 1.

Notice that this setting is written nowhere. That means that after a reboot of the virtual machine, you have to apply this setting again.

The last thing that you can do to manage CPUs, is to change the number of CPUs that is assigned to a virtual machine. You can do this from Virtual Machine Manager, and with the xm vcpu-set command. For example, to change the number of CPUs that are allocated to domain 1 to 4 VCPUs, use:

```
xm vcpu-set 1 4
```

When using this command, you will notice that it doesn't work all the times. This is because the operating system in the virtual machine has to offer support for dynamically changing the amount of CPUs as well. Therefore, it makes more sense to change the amount of VCPUs that a machine can use in its configuration file so that you are sure that the changed setting is persistent across a reboot as well.

Summary

For performance optimization within virtual machines, changing memory and CPU allocations is an important task. In this article, I taught you why. You have also learned how to modify the priority of a virtual machine on a physical CPU.

About the author: Sander van Vugt is an author and independent technical trainer, specializing in Linux since 1994. Vugt is also a technical consultant for high-availability (HA) clustering and performance optimization, as well as an expert on SLED 10 administration.

Dig Deeper

- **People who read this also read...**
  - Managing Xen shared resources: Credit scheduler and Xen scheduler
  - Monitoring Xen network traffic and usage with network resources
  - The 10 best virtual infrastructure management tools
  - What's behind the Citrix XenServer surge?
  - Xen virtual machines: Management options

- **Related tags**
  - ddo
  - network
  - oem
  - raid
  - ram
  - sbcglobal net sign in
  - search
  - telephone area code lookup
  - url
  - vm
  - vsphere 5
  - what is internet
  - what is software
  - xen cpu allocation
  - xenserver
- **Related glossary terms**

  Terms for Whatis.com - the technology online dictionary
  - physical block device (PBD)
  - Xen

Show me more

This was first published in December 2007

Back to top

You May Also Be Interested In...

More Background

- XenApp, XenDesktop drive XenServer growth
- XenServer 5.6 adds new features

More Details

- A VMware admin's guide to the XenServer command line
- XenServer management for the VMware administrator

Search More Tips on Virtual Implementation

- News
  - **Latest Headlines**

- XenServer 6.0 beta out: Virtualization news in brief
- Open Virtualization Alliance unlikely to unseat VMware
- Jump-starting your data center with a virtualization strategy
- View All News

- Server Virtualization Topics

  - **Topics**

    - **Server virtualization management tools and practices**

      VM performance management, Capacity planning, Provisioning and configuration, Monitoring and troubleshooting virtual machines, P2V, V2V and V2P migration

      Server virtualization security management and compliance policies

      Security management, Server virtualization compliance and governance

      Server virtualization hypervisors and management

      VMware virtualization, VMware management tools, VMware conference coverage, Microsoft virtualization, VMware administration and how-tos, Microsoft Hyper-V management, Citrix XenServer, Open source virtualization, Emerging platforms, Vendor selection

    - **Server virtualization infrastructure and architecture**

      Servers and virtualization, Network virtualization, Virtualized clusters and high-performance computing, Cloud computing architecture, Cloud computing infrastructure, Application virtualization, Using virtual machine appliances, Virtualization how-tos and learning guides

      Virtualization backup and disaster recovery strategies

      Backup and disaster recovery, Virtual server backup and storage

      Server virtualization staffing and budgets

    - **Server virtualization strategies and use cases**

      Desktop virtualization, Lab, test and development, Cloud computing and virtualization strategies, Data center consolidation, Backup, disaster recovery and business continuity

      Benefits of server virtualization

      Server consolidation and improved resource utilization, Green data center: Reducing power consumption with virtualization, Reducing IT costs with server virtualization, Improving server management with virtualization

Challenges of server virtualization

Virtualization costs, licensing and support issues, Virtualization security and patch management, Downtime and data loss in virtualized environments, Preventing virtual machine sprawl

- **Hot Topics**
  - Vendor selection
  - Virtual server backup and storage
  - VM performance management

- Tutorials
  - **Advice & Tutorials**
    - Virtualization Decisions 2010 Purchasing Intentions Survey
    - Hyper-V vs. VMware guide
    - Private cloud computing tutorial: An introduction to private clouds
    - Control VM sprawl in your virtual server infrastructure
    - Virtualization Explained: Virtualization definitions you need to know
    - vSphere and vCenter licensing and pricing explained -- a VMWare license guide
  - **Technology Dictionary**
    - Find definitions and links to technical resources
    - Powered by WhatIs.com
- Expert Advice
  - **Tips**
    - Cloud management with Microsoft Concero: Battling VMware vCloud Director
    - Data deduplication reduces storage requirements: Law firm case study
    - DMZ design: Bringing virtualization and security admins together
    - View All Tips
  - **Answers**
    - VMware acquisitions: Moving beyond virtualization
    - IT career advice for graduates
    - Software licensing models need to get with the times
    - View All Answers
  - **Ask a Question**
    - Get help from our technical community
    - Powered by ITKnowledgeExchange.com
- White Papers
  - **Research Library**
    - White Papers

- Business Webcasts
- Downloads
- Powered by Bitpipe.com

- **Product Demos**

- Try out software demos
- Powered by 2020Software.com

- **Resource Centers**

- Server Virtualization: Finding the right server for your needs
- Virtualization for SMB University Center
- View All Resource Centers

- Blogs

- **Blogs**

- The Virtualization Room
- Powered by ITKnowledgeExchange.com

Search this site SEARCH Search

More from Related TechTarget Sites

- VMware
- Windows Server
- Cloud computing
- Virtual Desktop
- Data Center

- VMware

- **Patching VMware ESXi Installable**

Patching VMware ESXi Installable has changed since the release of ESXi 4.1. Now the only way to update the free hypervisor is with the vihostupdate command-line utility.

- **CapacityIQ: Installing VMware's capacity planning tool**

CapacityIQ adds an advanced reporting element to vCenter Server. By installing this capacity planning tool, you can capacity-plan for infrastructure expansion and control VM sprawl.

- **Five ways to increase your server consolidation ratio**

Bosses love server consolidation, because it reduces costs. To keep your bosses smiling, find new ways to increase your consolidation ratio.

- Windows Server
  - What happened to the disk cleanup utility in Windows Server 2008?

    Useful desktop applications like cleanmgr.exe are missing from the server editions of Windows. There is a way to get them back.
  - When is a Recycle Bin Not a Recycle Bin? When it's in Active Directory

    The Active Directory Recycle Bin isn't exactly the same as the one found on desktop operating systems. That doesn't mean it isn't useful.
  - Scripting: VMware PowerCLI vs. Microsoft PowerShell

    VMware's PowerCLI sits atop Microsoft's PowerShell, but does it make scripting on a virtualization platform better?
- Cloud computing
  - Federated cloud redux with Hexagrid

    Federated clouds are making more sense for cloud providers looking to connect their own data centers, versus promoting strategic partnerships and crossovers with other providers.
  - Sharding relational databases in the cloud

    When handling Big Data in the cloud, make sure that your cloud databases are equipped for the necessary sharding processes.
  - Is VMware vCloud Express in trouble?

    VMware's quest to partner with service providers for Amazon-style clouds seems to be coming apart due to a lack of serious interest and gratification for providers.
- Virtual Desktop
  - Virtual Desktop Storage Basics: Understanding storage requirements

    In the second segment of this four-part e-book, we'll help you understand your virtual desktop storage requirements and manage storage for optimal performance.
  - Analyzing virtual desktop benefits, efficient delivery options

    VDI provides desktop management benefits, but infrastructure costs cut into ROI. In part one of this two-part tip, we explore the pros and cons of virtual desktops and deployment options.

- **Microsoft virtual desktop licensing guide**

  Microsoft licensing is convoluted, particularly for VDI. This guide navigates Windows virtual desktop licensing, how to reduce licensing costs, and what's legal and what's not.

- Data Center

  - **Indian outsourcer picks up mainframe graveyard shift**

    An Indian company is offering outsourcing services for organizations looking to fill late-night mainframe slots.

  - **CA overhauls mainframe portfolio: News in brief**

    CA conducts annual mainframe software blitz; Fujitsu updates its four- and eight-socket Primergy x86 servers; and Sanbolic releases a new version if its Melio cluster file system.

  - **HP vs. Cisco hardware battle yields deep discounts for IT shops**

    As Hewlett-Packard and Cisco duke it out for new data center deals, IT shops get fire-sale prices on high-end hardware. Now, how much would you pay?